

# Girard's "Fixpoint Theorem"

Linear Logic Mailing List

19 February 1992

Lars Hallnäs

University of Borås

Peter Schroeder-Heister

University of Tübingen

2022

On 5 February 1992, Jean-Yves Girard made a contribution to the Linear Logic Mailing List<sup>1</sup> with the title “A Fixpoint Theorem in Linear Logic” (<https://www.seas.upenn.edu/~sweirich/types/archive/1992/msg00030.html>). There he presented a view of logic programming which was related to what we had been working on for some time, Hallnäs at the Swedish Institute of Computer Science (SICS) and Schroeder-Heister at the Universities of Konstanz and Tübingen<sup>2</sup>. Girard referred to our work in a way that we saw as misleading. We thus formulated a reply which appeared on the list on 19 February 1992 (<https://www.seas.upenn.edu/~sweirich/types/archive/1992/msg00043.html>). As it contains aspects that are still of interest, and as it has been occasionally referred to, we reproduce it here as an online resource with a DOI and thus a stable internet address.

Lars Hallnäs and Peter Schroeder-Heister

---

<sup>1</sup> [linear@cs.stanford.edu](mailto:linear@cs.stanford.edu), included in the TYPES e-mail forum which focuses on type theory and is still active today: <https://lists.seas.upenn.edu/mailman/listinfo/types-list>.

<sup>2</sup>For example, “A Proof-Theoretic Approach to Logic Programming”, *J. Log. Comp.* 1 (1990), 261–283, 635–660, DOI: [10.1093/logcom/1.2.261](https://doi.org/10.1093/logcom/1.2.261) and [10.1093/logcom/1.5.635](https://doi.org/10.1093/logcom/1.5.635), “Partial Inductive Definitions”, *Theor. Comp. Sci.* 87 (1991), 115–142, DOI:[10.1016/S0304-3975\(06\)80007-1](https://doi.org/10.1016/S0304-3975(06)80007-1), “Cut Elimination in Logics with Definitional Reflection” in: D. Pearce (ed.), *Nonclassical Logics and Information Processing*, Springer 1992, DOI:[10.1007/BFb0031929](https://doi.org/10.1007/BFb0031929).

# Girard's "Fixpoint Theorem"

---

- *To:* types
  - *Subject:* Girard's "Fixpoint Theorem"
  - *From:* [mssh01@mailserv.zdv.uni-tuebingen.de](mailto:mssh01@mailserv.zdv.uni-tuebingen.de) (Schroeder-Heister)
  - *Date:* Wed, 19 Feb 92 12:57:52 EST
  - *Sender:* [meyer@theory.lcs.mit.edu](mailto:meyer@theory.lcs.mit.edu)
- 

Date: Wed, 19 Feb 92 14:37:18 +0100  
To: [Linear@cs.stanford.edu](mailto:Linear@cs.stanford.edu)

Jean-Yves Girard's "A Fixpoint Theorem in Linear Logic"

In his contribution, Jean-Yves Girard mentions our work in a somewhat misleading way. We take this occasion to give some more detailed references.

1. The elimination rule for atoms which is the basis of Girard's work, has been proposed by us in a similar form since 1987 (for references see "A proof-theoretic approach to logic programming, II. Programs as definitions", JLC 1, 1990/91). As Girard's, our motivation was the reading of program-clauses of a logic program as definitional clauses. We called the new rule the rule of "definitional reflection" or "D-rule", thus emphasizing that when applying the rule one reflects on the fact that the clauses for introducing an atom according to the program exhaust the means for introducing that atom. We also called it the "P-left" or "(P /-)"-rule ("P-right" for the rule that corresponds to the resolution rule), emphasizing what we called "computational symmetry":

The P-left-rule allows

one to introduce an atom on the left of the turnstile rather than only on the right (the latter case, which is the standard one, corresponds to resolution, the novel former one to a principle dual to resolution). As an alternative theory of inductive definitions, where one deviates from the standard monotone case and its least fixpoint interpretation, our approach was developed under the heading "partial inductive definitions" (see TCS 87, 1991).

2. A logic programming language called GCLA is being developed at SICS. It is based on that theory and incorporates the rule of definitional reflection, including the computation of appropriate substitutions in the case of the D-rule. It particularly allows for bindings to variables in hypothetical or negative queries, and also for evaluation of function definitions based on the D-rule.

3. We have basically worked with intuitionistic (not classical!) logic. In this system, Cut-elimination indeed fails, if the bodies of program clauses contain implications. We are aware that for a contraction-free system Cut is obtained. Our proof uses induction over the triple <number of occurrences of the D-rule above a (generalized version of a) Cut, degree of Cut formula, length of derivations of Cut premisses> (appearing in "Non-Classical Logics and Information Processing", ed. D. Pearce, Springer LNAI).

4. We do not understand why deductive power should decrease in principle in the absence of Cut. We consider it basically a pragmatic matter whether in a certain situation one wants to have contraction or not, so one should be able to choose various structural frameworks in

which to work. The control language of GCLA is actually strong enough to permit such a choice to a certain extent, so contraction and consequently cut-elimination is not a matter of principle but is up to the choice of the programmer. When dealing with function definitions in our system, we actually work in the contraction-free variant.

5. As to the Girard's "fixpoint theorem", this is the basic criterion for any interpretation of a set of formulae as the definition of an atom  $p(x)$ . Since we do not assume cut as a primitive rule of inference, it is trivially true.

6. There seem to be a lot of examples that can be naturally interpreted in terms of the duality between the P-right and P-left rule. Proving that an object satisfies an inductively defined predicate relies on the P-right rule while evaluating a function with respect to a given definition on P-left. There is also the duality between induction and co-induction. Also, if you want to make a distinction between induction (global) and logical elimination rules (local, the P-left rule).

7. If we use the notion of higher level-rules we can define first order predicate logic by reading the introduction rules of natural deduction as definitional clauses. This process can then be iterated by adding definitions to predicate logic, arithmetic . . . . naive set theory. Through the Curry-Howard interpretation one sees that computationally all this means adding recursive data-types to a given lambda calculus, simply typed lambda calculus, system F, the lambda calculus you get from naive set theory with extensionality . . . . The other way round one sees that there is a logic behind these computational theories.

So where is the true foundation of all this?

The rule dual to resolution (the elimination of an atom with respect to a definition) seems to be a simple and natural candidate to be included in an interpretation of a set of formulas as a definition. Of course, it remains to explain the interpretation of the formulas themselves. Linear logic, BCK logic, some kind of partial interpretation . . . . Why is cut primitive? Why is cut not primitive? Why is contraction primitive? Why is contraction not primitive? What is silly and what is not? It is perhaps a little bit early to judge. . . .

Lars Hallnaes and Peter Schroeder-Heister.

- 
- Prev: [\*\*paper on geometry of interaction\*\*](#)
  - Next: [\*\*Linear logic semantics \(latex file\)\*\*](#)
  - Index(es):
    - [\*\*Main\*\*](#)
    - [\*\*Thread\*\*](#)